



Tema 2. Algoritmi: caracteristici, reprezentare, implementare

Obiective

- să recunoști un algoritm
- să identifice caracteristicile algoritmilor
- să identifice etapele algoritmilor

Fișa de documentare 2.1. Etapele rezolvării problemelor. Algoritmi. Caracteristicile algoritmilor.



Algoritmi

Noțiunea de algoritm este prezentă azi în contexte diferite. Termenul algoritm vine de la numele matematicianului persan Abu Ja'Far Mohamed ibn Musa al Khwarizmi (circa 825 e.n.), care a scris o carte cunoscută sub denumirea latină de "Liber algorithmi". Tot el a introdus denumirea de "algebra" în matematică.

În trecut, termenul de algoritm era folosit numai în domeniul matematicii, însă datorită dezvoltării calculatoarelor, astăzi "gândirea algoritmică" nu mai este un instrument specific matematicii ci folosit în diverse domenii.



*Prin algoritm înțelegem o succesiune **finită** de operații cunoscute care se execută într-o **succesiune logică bine stabilită** astfel încât plecând de la un set de date de intrare, să obținem într-un interval de **timp finit** un set de date de ieșire.*



Caracteristicile algoritmilor



Finitudine – proprietatea algoritmilor de a furniza datele de ieșire într-un timp finit (adică după un număr finit de pași).

De exemplu, dacă avem următoarea problemă: Se citește un număr n natural. Să se efectueze operația de extragere a radicalului și să se afișeze rezultatul. Această problemă nu este un proces finit, deoarece nu s-a specificat precizia cu care se va furniza rezultatul.



Claritatea - algoritmul trebuie să descrie operațiile clar și fără ambiguități.



Generalitatea – proprietatea algoritmilor de a rezolva o întreagă clasă de probleme de același fel.

De exemplu adunarea $2+8$ este o problemă care adună numai aceste două numere, însă dacă elaborăm o metodă de rezolvare care va aduna $a+b$, unde a și b pot avea orice valori întregi, spunem că am realizat un algoritm general.



Corectitudinea – spunem că un algoritm este corect dacă el furnizează în mod corect datele de ieșire pentru toate situațiile regăsite în datele de intrare.

De exemplu, trebuie să evaluăm expresia $E=a/b+c$. O succesiune de pași pentru evaluarea expresiei este:

- se citește a, b, c
- se calculează a/b , apoi rezultatul se adună cu c .
- se atribuie lui E valoarea calculată
- se afișează E

Acest algoritm NU furnizează rezultatul corect pentru toate valorile de intrare. În cazul în care $b=0$, împărțirea nu se poate efectua dar algoritmul nu verifică acest lucru.

Există totuși algoritmi care sunt corecți, clari, generali și furnizează soluția într-un timp finit însă mai lung sau folosesc mai multă memorie decât alți algoritmi. Aceasta înseamnă că atunci când elaborăm un algoritm, nu ne oprim la prima soluție găsită. Vom încerca să găsim algoritmi care să dea soluția într-un timp cât mai scurt, cu cât mai puțină memorie folosită. Cu alte cuvinte vom încerca să elaborăm algoritmi **eficienți**.



Numim deci **eficiență** - capacitatea algoritmului de a da o soluție la o problemă într-un timp de execuție cât mai scurt, folosind cât mai puțină memorie.



Etapele rezolvării problemelor



Rezolvarea unei probleme este un proces complex, care are mai multe etape.

1. Analiza problemei, pentru a stabili datele de intrare și de ieșire.
2. Elaborarea unui algoritm de rezolvare a problemei.
3. Implementarea algoritmului într-un limbaj de programare.
4. Verificarea corectitudinii algoritmului implementat.
5. Analiza complexității algoritmului.

ETAPELE REZOLVĂRII UNEI PROBLEME



Toate aceste etape vor fi evidențiate pe algoritmii ce vor fi prezentați în fișele de documentare următoare.

Etapele rezolvarii unei probleme

Rezolvarea unei probleme constituie un proces complex, care comporta mai multe etape.

1. **Analiza problemei** in scopul stabilirii datelor de intrare, precum si a rezultatelor pe care trebuie sa le obtinem prin rezolvarea problemei.

2. **Elaborarea unui algoritm** de rezolvare a problemei.

3. **Implementarea algoritmului** intr-un limbaj de programare.

4. **Verificarea corectitudinii** algoritmului propus.

Un prim pas consta in testarea programului pe diverse seturi de date de test.

Seturile de date de test trebuie elaborate cu atentie, astfel incat sa acopere, pe cat posibil, toate variantele de executie a algoritmului, inclusiv situatii de exceptie, si sa verifice daca fiecare subproblema a problemei date este rezolvata corect (daca este posibil, se va testa separat fiecare modul de program).

Testarea poate pune in evidenta, eventual, omisiuni sau erori de concepie a algoritmilor, dar nu garanteaza corectitudinea algoritmului. Pentru aceasta ar trebui sa testam algoritmul pe toate seturile posibile de date de intrare, ceea ce este practic imposibil. Din acest motiv se impune utilizarea unor metode formale de demonstrare a corectitudinii algoritmului, etapa de obicei deosebit de laborioasa, necesitand un aparat matematic complex.

5. **Analiza complexitatii** algoritmului.

In general, exista mai multi algoritmi de rezolvare a unei probleme date. Pentru a alege cel mai bun algoritm, trebuie sa analizam acesti algoritmi in scopul determinarii eficientei lor si, pe cat posibil, a optimalitatii lor.

Eficienta unui algoritm se evalueaza din doua puncte de vedere:

1. **din punctul de vedere al spatiului de memorie** necesar pentru memorarea valorilor variabilelor care intervin in algoritm;

2. **din punctul de vedere al timpului de executie.**

Complexitatea spatiu o vom analiza cu precadere atunci cand vom transpune algoritmul intr-un limbaj de programare.

Pentru a estima complexitatea timp vom presupune ca se lucreaza pe un calculator «clasic», in sensul ca o singura instructiune este executata la un moment dat. Astfel, timpul necesar executiei programului depinde de numarul de operatii elementare efectuate de algoritm.

ETAPELE DE REZOLVARE A UNEI PROBLEME, DE LA ANALIZA LA EXECUTIE

